# React js

Lecture 5

# Agenda

- Recap last lecture points
- What is Redux ?
- Redux components
- Getting started with store
- Context
- Redux vs Context

# Redux

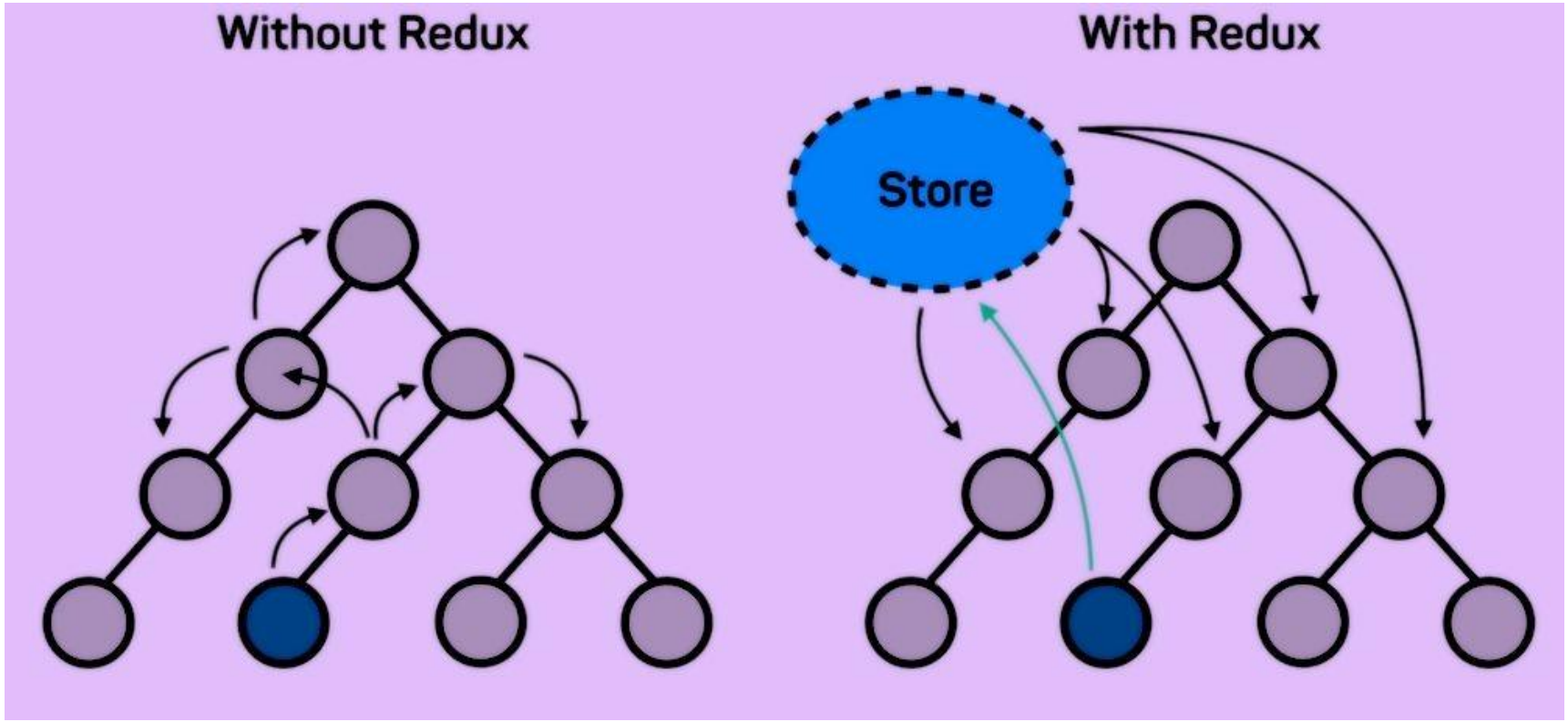
Redux is a predictable state container and the state of your application is kept in a store, and each component can access any state that it needs from this store.

https://redux.js.org/introduction/getting-started

To use Redux in your react app you need to install it first :

```
npm install redux&toolkit react-redux
```

# Redux

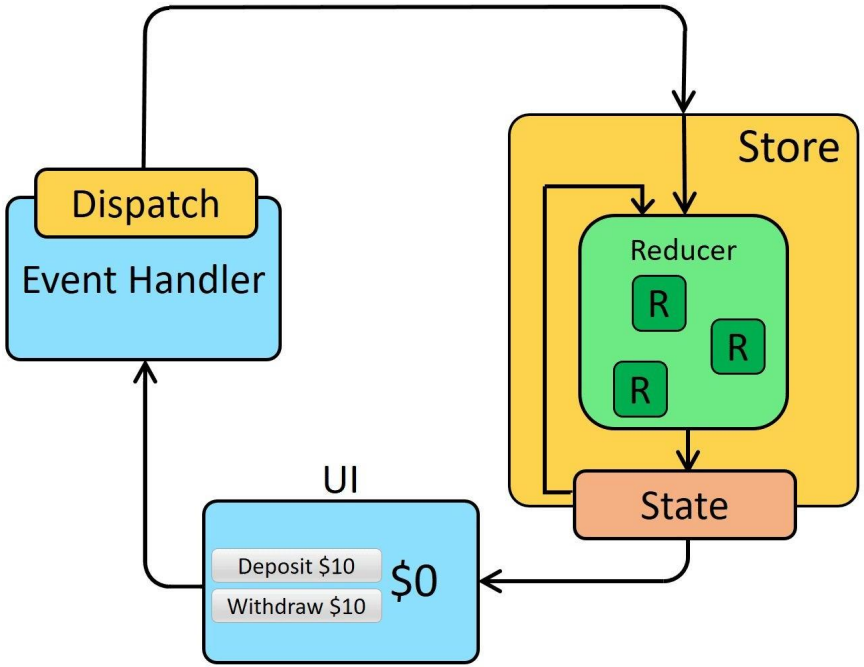


# How Redux works

There is a central store that holds the entire state of the application. Each component can access the stored state without having to send down props from one component to another.

**There are three building parts:**

- actions
- store
- reducers



# How Redux works : Actions

Actions are plain JavaScript objects, and they must have a **type** property to indicate the type of action to be carried out.

Example :

```
const setUsername = (payload) => {

  return {

    type: "LOGIN",

    payload: payload

  }

}
```

# How Redux works : Reducers

Reducers are pure functions that take the current state of an application, perform an action, and return a new state. These states are stored as objects, and they specify how the state of an application changes in response to an action sent to the store.

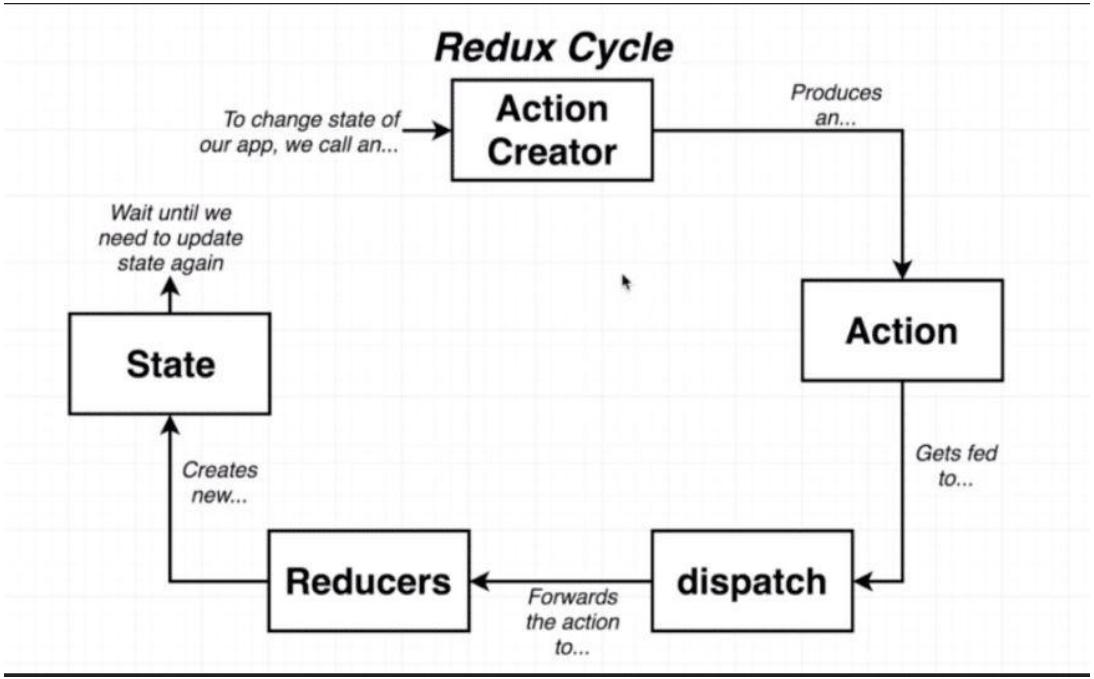
# How Redux works :

```
import * as types from "./types";

export default (state = {}, action) => {
  switch (action.type) {
    case type:
      return {
        ...state,
        ...action.payload,
      };
    default:
      return state;
  }
};
```

# How Redux works : Store

The store holds the application state. There is only one store in any Redux application.



# How Redux

```
import { createStore} from "redux";
import reducers from "./reducers";

//redux dev tool
const composeEnhancers =
   window.     REDUX_DEVTOOLS_EXTENSION  && window.    REDUX_DEVTOOLS_EXTENSION
   ();
const store = createStore(reducers , composeEnhancers  );

export default store;
```

Or install redux-devtools-extension

```
import { composeWithDevTools } from 'redux-devtools-extension'
```

Then pass composeWithDevTools() instead of composeEnhancers

# How Redux works : Wrap application with Redux

```
 <Provider
 store={store}>

  <App />

</Provider>
```

# UseSelectoR()

To read and get different store values

Syntax : const state = useSelector(state => state)

# UseDispatch()

To Update store values and dispatch actions  Syntax : const dispatch = useDispatch();

```
dispatch(action());
```

# Context

Context provides a way to pass data through the component tree without having to pass props down manually at every level.

## When to Use Context

Context is designed to share data that can be considered “global” for a tree of React components, such as the current authenticated user, theme, or preferred language.

# Context VS Redux

- If you are using Redux only to avoid passing props down to deeply nested components, then you could replace Redux with the Context API. It is exactly intended for this use case.

- On the other hand, if you are using Redux for everything else (handling your application's logic outside of your components, centralizing your application's state, using Redux DevTools to track when, why, and how your application's state changed, or using plugins such as Redux Form, Redux Saga, etc…), then there is absolutely no reason for you to abandon Redux.

  The Context API doesn't provide any of this.

# Context : CReate context

**Create context file config :**

```
import React from "react";

const contextFeature = React.createContext();
export const contextProvider= contextFeature.Provider;

export default contextFeature ;
```

# Context : Use context

- **Wrap the components that needs to have the provider values with context provider:**

```
const [value, setValue] = useState(initial value);
<contextProvider   value={{ value, setValue}}>
     <clild />
</contextProvider>
```

- **In child component you can use and set context value using useContext hook**

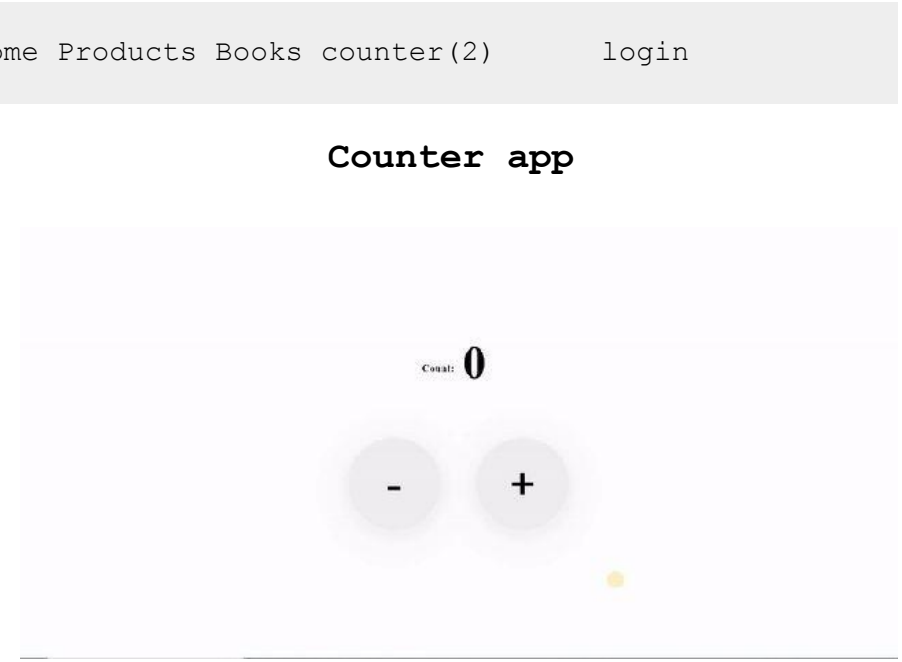
```
const {value , setValue} = useContext(contextFeature)
```

# Task :CounteR App

Create a new page called counter with the following requirements :

1. Add link to navbar with counter page and counter number should be updated in navbar with counter
2. Counter component with + / - and reset counter number , make sure that counter values not less than 0

Please use redux cycle with counter to read and update counter number from store

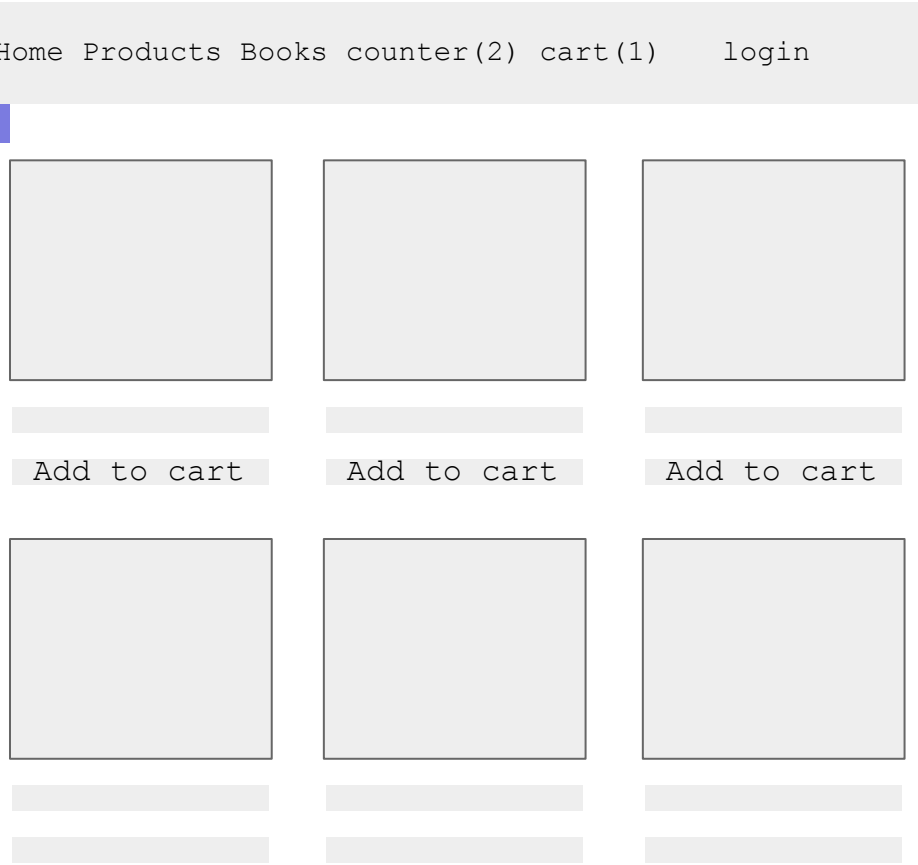


# ADD To CaRt

Continue on moives page ,

Add functionality add to cart and show cart counter in navber.

And make cart icon clickable to show cart items in cart page with quantity and options to increase or decrease item quantity



# ResouRces

React links :

- https://www.youtube.com/watch?v=Dorf8i6lCuk
- https://www.youtube.com/watch?v=w7ejDZ8SWv8&t=2559s
- https://www.youtube.com/watch?v=0riHps91AzE
- Chat with firebase https://www.youtube.com/watch?v=zQyrwxMPm88
- Medhat Dawoud Channel : https://www.youtube.com/channel/UCuwTHYdMavwEPsZ6OAkXfig
- Mosh channel : https://www.youtube.com/channel/UCWv7vMbMWH4-V0ZXdmDpPBA
- Formik in deep : https://www.youtube.com/watch?v=a94FOvaBomQ&list=PLC3y8-rFHvwiPmFbtzEWjESkqBVDbdgGu